Seminar Nasional Aplikasi Teknologi Informasi 2012 (SNATI 2012) *ISSN: 1907-5022*
Yogyakarta, 15-16 Juni 2012

ALGORITMA ENKRIPSI CITRA DIGITAL DENGAN KOMBINASI DUA *CHAOS MAP* DAN PENERAPAN TEKNIK SELEKTIF TERHADAP BIT-BIT *MSB*

Rinaldi Munir[1]
[1]*Sekolah Teknik Elektro dan Informatika (STEI), Institut Teknologi Bandung (ITB)*
Jl.Ganesha 10, Bandung 40132
E-mail:rinaldi-m@stei.itba.c.id

ABSTRAK

Di dalam makalah ini dipresentasikan sebuah algoritma enkripsi citra berbasis chaos dan penggunaan teknik enkripsi selektif untuk meminimumkan volume komputasi. Dua buah pemetaan chaos digunakan, yaitu Arnold Cat Map dan Logistic Map. Sebelum dienkripsi, citra diacak dengan Arnold Cat Map, selanjutnya teknik enkripsi selektif diterapkan dengan memilih hanya empat bit MSB dari setiap pixel untuk di-XOR-kan dengan keystream yang dibangkitkan dari Logistic Map. Hasil ekperimen memperlihatkan penggunaan kedua fungsi chaos tersebut dapat menghasilkan efek confusion dan diffusion, dan penggunaan teknik enkripsi selektif hanya memproses 50% dari keseluruhan data citra. Citra hasil enkripsi memperlihatkan histogram yang terdistribusi relatif uniform sehingga menyulitkan penyerang melakukan analisis statistik untuk menemukan kunci atau plain-image. Sensitifitas chaos memperlihatkan bahwa algoritma ini aman dari exhaustive-key search attack.

Kata Kunci: enkripsi, citra, chaos, selektif, Arnold Cat Map, Logistic Map

1. PENDAHULUAN

Citra (*image*) merupakan salah satu bentuk data atau informasi yang disajikan secara visual. Citra memainkan peranan penting dalam industri multimedia saat ini. Citra juga merupakan unsur pembentuk video, sebab sebuah video pada dasarnya disusun oleh rangkaian *frame* citra yang ditampilkan dalam tempo yang cepat.

Selain disimpan di dalam media *storage*, citra juga dikirim secara elektronis melalui saluran publik seperti internet. Penyimpanan atau pengiriman citra melalui saluran transmisi rawan terhadap pengaksesan atau penyadapan oleh pihak yang tidak memiliki otoritas. Oleh karena itu, untuk melindungi kerahasiaan citra dari pengaksesan ilegal, maka enkripsi citra telah digunakan secara luas sebagai salah satu cara menjaga keamanan informasi.

Kebanyakan algoritma kriptografi yang tersedia ditujukan untuk mengenkripsi data teks. Algoritma kriprografi konvensional seperti *DES*, *AES*, *Blowfish*, *RC4*, *RSA*, dan lain-lain tidak mangkus digunakan untuk mengenkripsi data citra. Hal ini karena data citra umumnya bervolume sangat besar dibandingkan dengan data teks, sehingga diperlukan waktu komputasi yang lebih lama untuk mengenkripsi citra. Untuk kebutuhan aplikasi yang real-time seperti *teleconference*, *live video streaming*, dan lain-lain, jelas algoritma konvensional tidak cocok untuk mengenkripsi citra.

Banyak peneliti yang telah mengembangkan algoritma enkripsi untuk citra digital. Menurut Younes (2008), kebanyakan algoritma enkripsi citra dapat dikelompokkan menjadi dua golongan: (a) algoritma enkripsi selektif non-*chaos*, dan (b) algoritma enkripsi selektif atau non-selektif yang berbasis *chaos*. Algoritma selektif artinya hanya mengenkripsi sebagian elemen di dalam citra namun efeknya mengenkripsi citra secara keseluruhan. Algoritma enkripsi selektif bertujuan meminimalkan volume komputasi selama proses enkripsi dan dekripsi sehingga ia dapat digunakan untuk kebutuhan aplikasi yang *real-time*.

Adapun *chaos* menjadi topik yang atraktif di dalam kriptografi karena tiga alasan: (1) sensitivitas terhadap kondisi awal, (2) berkelakuan acak, dan (3) tidak memiliki periode berulang. Penggunaan *chaos* di dalam kriptografi dapat menghasilkan efek *diffusion* seperti yang dinyatakan oleh Shanon (Schneier, 1996). Beberapa algoritma enkripsi citra yang menggunakan *chaos* antara lain (Xiang, 2007), (Zhang, 2006), (Jolfaei, 2010), (Struss, 2009), dan Fu (2012).

Chaos di dalam kriptografi umumnya digunakan sebagai pembangkit bilangan acak. Bilangan-bilangan acak itu digunakan sebagai *keystream* (dengan operasi XOR sederhana) atau untuk mengacak susunan *pixel* di dalam citra. Barisan bilangan acak dibangkitkan dengan sebuah fungsi *chaos* (*map*). Xiang (2007) menggunakan *Tent Map* sebagai pembangkit kunci enkripsi, Struss (2009) dan Zhang (2006) menggunakan *Arnold Cat Map* untuk mengacak *pixel-pixel*. Hal yang sama juga dilakukan oleh Jolafel (2010) tetapi menggunakan *Henon Map* untuk permutasi *pixel-pixel* sebelum dienkripsi dengan *stream cipher*, sedangkan Fu (2012) mengkolaborasikan *Chebysev Map* sebagai pembangkit *keystream*.

Di dalam makalah ini diusulkan sebuah algoritma enkripsi citra yang berbasis *chaos*. Algoritma tersebut mengkombinasikan dua buah fungsi *chaos* yaitu *Arnold Cat Map* dan *Logistic Map*. *Arnold Cat Map* digunakan untuk mengacak

susunan *pixel-pixel*, sedangkan *Logistic Map* digunakan sebagai pembangkit *keystream*. Untuk menghemat volume komputasi selama proses enkripsi/dekripsi, teknik enkripsi selektif yang diusulkan oleh Xiang (2007) diterapkan dengan hanya meng-XOR-kan *keystream* dengan bit-bit *MSB* yang berperan menentukan persepsi visual terhadap obyek di dalam citra.

2. CHAOS

Karakteristik utama sistem *chaos* adalah sensitivitasnya terhadap parameter nilai awal (*initial value*). Sensitivitas ini berarti jika *chaos map* diiterasikan sejumlah kali, maka perubahan kecil pada parameter nilai awal *map* menghasilkan perbedaan yang signifikan pada nilai fungsinya. Karakteristik ini penting di dalam kriptografi sebab bersesuaian dengan prinsip *diffusion* yang dikemukakan oleh Shannon (Schneier, 1996). Dengan prinsip *diffusion* ini maka pengubahan satu bit parameter nilai awal *chaos* dapat menyebabkan cipherteks tidak bisa diprediksi lagi dan sebagai konsekuensinya cipherteks tetap tidak dapat didekripsi. Dua buah fungsi *chaos* yang digunakan di dalam algoritma ini adalah *Arnold Cat Map* dan *Logistic Map*. Masing-masing fungsi dijelaskan pada sub-bagian di bawah ini.

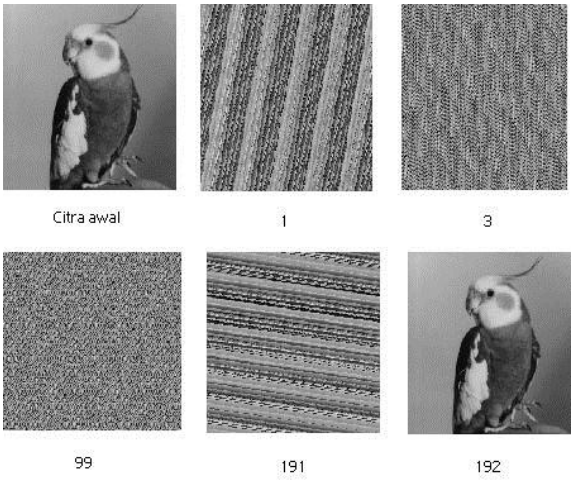
2.1 Arnold Cat Map

Arnold Cat Map (*ACM*) adalah fungsi *chaos* dwimatra yang mentransformasikan koordinat (*x*, *y*) di dalam citra ke koordinat baru di dalam citra yang sama. Persamaan transformasinya adalah

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & p \\ q & pq+1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \bmod(N) \qquad (1)$$

yang dalam hal ini (x_i, y_i) adalah posisi *pixel* di dalam citra berukuran $N \times N$ dan (x_{i+1}, y_{i+1}) posisi *pixel* yang baru setelah transformasi; *p* dan *q* adalah *integer* positif. Determinan $\begin{bmatrix} 1 & p \\ q & pq+1 \end{bmatrix}$ harus sama dengan 1 agar hasil transformasianya tetap berada di dalam area citra yang sama (*area-preserving*). *ACM* adalah pemetaan satu-ke-satu yang berarti setiap posisi *pixel* ditransformasikan ke posisi lain secara unik (Yu, 2006). Fungsi *chaos* ini ditemukan oleh Vladimir Arnold pada tahun 1960 yang menggunakan citra seekor kucing sebagai eksperimennya.

Iterasi *ACM* terhadap sebuah citra akan mengacak citra tersebut, yang sama artinya dengan mengenkripsi citra. Dengan melakukan iterasi berkali-kali diperoleh citra acak yang berbeda-beda. Namun, sesudah iterasi tertentu kembali dihasilkan citra awal semula, oleh karena itu *ACM* memiliki perioda. Gambar 1 memperlihatkan iterasi *ACM* terhadap citra 'burung'.



Gambar 1. Iterasi ACM pada citra 'burung'

Jumlah iterasi yang diperlukan agar citra acak kembali ke citra semulal berbeda-beda bergantung pada ukuran citra. Menurut *paper* yang ditulis Freeman J. Dyson dan Harold Falk jumlah iterasi yang diperlukan kurang dari 3*N*, yang dalam hal ini *N* adalah dimensi citra (misalnya jika citra berukuran 256×256 maka $N = 256$) (Struss, 2009).

Meskipun *p* dan *q* adalah paramater rahasia, namun karena sifat periodik *ACM* yang dapat menghasilkan kembali citra semula, maka enkripsi dengan menggunakan *ACM* saja tidak aman, sebab melalui *hack* sederhana nilai *p* dan *q* dapat ditemukan melalui operasi *brute force*. Selain itu *ACM* hanya mengubah posisi *pixel* di dalam citra tetapi tidak mengubah nilai *pixel*. Oleh karena itu, kita perlu melakukan *encoding* nilai-nilai *pixel* dengan menggunakan *chaos map* yang kedua, yaitu *Logistic Map*.

2.2 Logistic Map

Logistic Map adalah fungsi *chaos* satu matra yang berbentuk

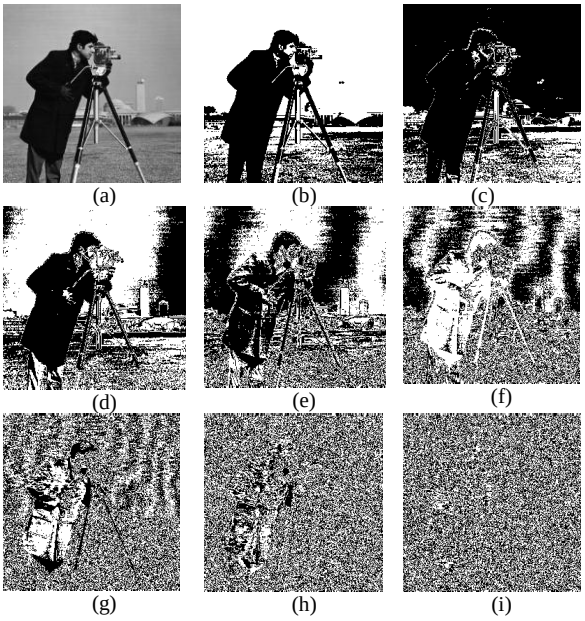
$$x_{i+1} = r\, x_i\, (1 - x_i) \qquad (2)$$

Nilai x_i adalah antara 0 dan 1. Konstanta *r* menyatakan laju pertumbuhan fungsi dan $0 \le r \le 4$. Persamaan ini mulai menuju sifat chaotik ketika $r > 3.75$. Ketika $r = 4$ fungsi secara total menjadi *chaos* dan nilai-nilai x_i tidak dapat diprediksi lagi yang berarti sudah acak. Nilai awal *chaos*, x_0, dan konstanta *r* berperan sebagai parameter rahasia *Logistic Map*. Nilai-nilai acak yang dihasilkan tidak mempunyai periode, meskipun demikian *Logistic Map* tetap deterministik.

Barisan nilai acak yang dihasilkan dari iterasi *Logitstic Map* sensitif terhadap perubahan kecil nilai awal. Dengan mengubah nilai x_0 sedikit saja sebesar Δ (yaitu $x_0 + \Delta$), barisan nilai acak yang dihasilkan – setelah diiterasi beberapa kali-- berbeda signifikan dengan barisan acak sebelumnya dengan nilai awal x_0.

3. ENKRIPSI SELEKTIF

Setiap *pixel* direpresentasikan dalam sejumlah *byte*. Susunan bit pada setiap *byte* adalah $b_7b_6b_5b_4b_3b_2b_1b_0$. Bit-bit paling kiri adalah bit paling berarti (*most significant bits* atau *MSB*), sedangkan bit-bit paling kanan adalah *least significant bits* atau *LSB*. Jika setiap bit ke-*i* dari setiap *pixel* pada citra *grayscale* diekstraksi dan diplot ke dalam setiap *bitplane image* maka kita memperoleh delapan buah citra biner. Gambar 2(a) sampai 2(i) memperlihatkan *bitplane image* dari citra *cameraman*. Gambar 2(b) hingga 2(f) yang diambil dari bit-bit *MSB* masih dapat memperlihatkan wujud objek di dalam citra tetapi dari Gambar 1(g) hingga 1(i) yang diambil dari bit-bit *LSB* sudah terlihat seperti citra acak.



Gambar 2. Delapan *Bitplane* pada citra *cameraman*

Karena bit-bit *MSB* menentukan rupa obyek di dalam citra, maka dengan mengubah bit-bit *MSB* tersebut dihasilkan citra yang sudah tidak dapat dikenali lagi. Jadi, kita cukup memilih hanya bit-bit *MSB* saja yang dienkripsi sebab dengan hanya mengenkripsi bit-bit tersebut maka keseluruhan citra menjadi tidak dapat dikenali lagi. Inilah yang mendasari teknik enkripsi selektif berbasis bit-bit *MSB*.

Xiang (2007) meneliti bahwa untuk memperoleh keseimbangan antara tingkat keamanan dan pertimbangan performansi komputasi, maka enkripsi empat bit *MSB* (yaitu $b_7b_6b_5b_4$) merupakan pemilihan yang optimal. Dengan mengenkripsi hanya 4-bit *MSB* berarti kita hanya perlu mengenkripsi 50% saja dari keseluruhan citra untuk memperoleh citra terenkripsi namun tingkat keamanannya tetap terjamin.

Di dalam makalah ini, 4-bit *MSB* dari setiap *pixel* dienkripsi seperti *stream cipher* dengan menggunakan operasi XOR sederhana:

$$c_i = p_i \oplus k_i \qquad (3)$$

Untuk dekripsi digunakan persamaan kebalikan,

$$p_i = c_i \oplus k_i \qquad (4)$$

Pada persamaan (3) dan (4) di atas p_i adalah 4-bit *MSB* dari suatu *pixel* pada *plain-image*, c_i adalah 4-bit *MSB* suatu *pixel* pada *cipher-image*, dan k_i adalah *keystream* 4-bit yang dibangkitkan dari *Logistic Map*.

4. USULAN ALGORITMA

Algoritma enkripsi citra yang diusulkan di dalam makalah dapat digunakan baik untuk citra *grayscale* maupun untuk citra berwarna. Citra yang dienkripsi harus berukuran bujursangkar ($N \times N$) agar *ACM* dapat diterapkan. Jika citra tidak berukuran bujursangkar (yaitu $M \times N$), maka harus ditambahkan sejumlah baris atau kolom dengan *pixel-pixel* bernilai 0 sedemikian sehingga ukuran citra menjadi bujursangkar.

Secara garis besar algoritma enkripsi terdiri dari dua bagian:

(a) Pengacakan *pixel-pixel* citra dengan *ACM*;
(b) Enkripsi *stream cipher*, yaitu operasi XOR antara 4-bit *MSB* dari setiap *pixel* dengan 4-bit *keystream*.

Algoritma dekripsi adalah kebalikan dari enkripsi, dimulai dengan langkah (b) kemudian langkah (a).

4.1 Pembangkitan *Keystream*

Bit-bit *MSB* yang dipilih dari setiap *pixel* di-XOR-kan dengan *keystream* yang panjangnya empat bit. Empat-bit *keystream* k_i diperoleh sebagai berikut: nilai *chaos* x_i dikalikan dengan 10 berulangkali sampai ia mencapai panjang angka (*size*) yang diinginkan, selanjutnya potong hasil perkalian tersebut untuk mengambil bagian *integer*-nya saja. Secara matematis, nilai *chaos x* dikonversi ke *integer* dengan menggunakan persamaan berikut (Lampton, 2002):

$$T(x, size) = \left\| x * 10^{count} \right\|, x \neq 0 \qquad (5)$$

yang dalam hal ini *count* dimulai dari 1 dan bertambah 1 hingga $x * 10^{count} > 10^{size-1}$. Hasilnya kemudian diambil bagian *integer* saja (dilambangkan dengan pasangan garis ganda pada persamaan 5 yang melambangkan *truncation*). Empat bit terakhir dari representasi biner *integer* tersebut dijadikan sebagai k_i.

Tanpa kehilangan generalisasi, berikut ini dijelaskan langkah-langkah di dalam algoritma enkripsi untuk citra *grayscale*.

4.2 Enkripsi

Input: citra awal *P* (*plain-image*), *p*, *q*, *m* (jumlah iterasi *ACM*), *r*, x_0
Output: citra terenkripsi *C* (*cipher-image*)

Langkah 1.) Lakukan permutasi, yaitu mengacak *pixel-pixel* di dalam citra *P* dengan mengiterasikan *ACM* sejumlah *m* kali.

Langkah 2.) Ekstraksi 4-bit *MSB* setiap *pixel* dari citra hasil langkah 1 di atas, nyatakan setiap 4-bit tersebut sebagai p_i (i = 1,2, .. n). Catatan: $n = N \times N$.

Langkah 3.) Iterasikan *Logistic Map* untuk memperoleh nilai-nilai *keystream* sesuai dengan algoritma di dalam 4.1.

Langkah 4.) Enkripsi p_i dengan k_i menggunakan persamaan (3) menghasilkan c_i.

Langkah 5) c_1, c_2, …, c_n selanjutnya menggantikan 4-bit *MSB* dari setiap *pixel* yang dienkripsi. Hasil enkripsi terhadap seluruh *pixel* adalah citra terenkripsi (*cipher-image*), *C*.

4.3 Dekripsi

Input: citra terenkripsi *C* (*cipher-image*), *p*, *q*, *m* (jumlah iterasi *ACM*), *r*, x_0
Output: citra semula *P* (*plain image*)

Langkah 1.) Ekstraksi 4-bit *MSB* setiap *pixel* dari *cipher-image C*, nyatakan setiap 4-bit tersebut sebagai c_i (i = 1,2, .. n). Catatan: $n = N \times N$.

Langkah 2.) Iterasikan *Logistic Map* untuk memperoleh nilai-nilai *keystream* sesuai dengan algoritma di dalam 4.1.

Langkah 3.) Dekripsi c_i dengan k_i menggunakan persamaan (4).

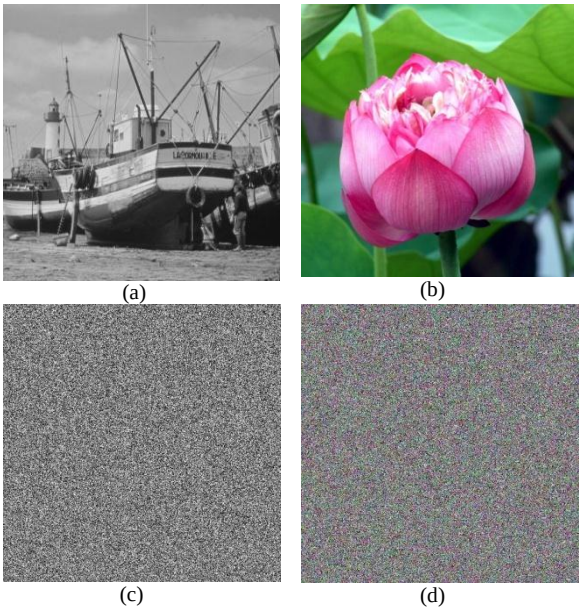
Langkah 4.) p_1, p_2, …, p_n selanjutnya menggantikan 4-bit *MSB* dari setiap *pixel* yang didekripsi.

Langkah 5.) Lakukan *inverse permutation*, yaitu menyusun kembali *pixel-pixel* citra hasil dari langkah 4 ke susunan semula dengan mengiterasikan *ACM* sejumlah *m* kali. Hasil *inverse permutation* ini adalah citra semula (*plain-image*), *P*.

Algoritma enkripsi/dekripsi di atas dapat dirampatkan untuk citra berwarna yang mana setiap *pixel* memiliki komponen *red* (*R*), *green* (*G*), dan *blue* (*B*). Prosesnya enkripsinya dilakukan tiga kali, masing-masing untuk kanal *R*, *G*, dan *B*. Jadi, dari setiap kanal warna diambil 4-bit *MSB* kemudian dioperasikan dengan algoritma di atas secara terpisah untuk masing-masing kanal. Pengacakan *pixel-pixel* dengan *ACM* juga dilakukan masing-masing untuk setiap kanal warna.

5. EKSPERIMEN DAN PEMBAHASAN

Untuk mengetahui fungsionalitas dan keamanan algoritma, maka dilakukan eksperimen dengan menggunakan kakas Matlab. Dua buah citra uji yang digunakan adalah sebuah citra *grayscale* dan sebuah citra berwarna. Kedua buah citra tersebut adalah citra 'kapal' (512 × 512) dan citra 'bunga' (512 × 512), seperti ditunjukkan pada Gambar 3(a) dan 3(b). Parameter kunci yang dipakai di dalam eksperimen adalah: p = 15, q = 27, r = 3.98, x_0 = 0.6938, dan m = 5. Citra hasil enkripsi (*cipher-image*) masing-masing diperlihatkan pada Gambar 3(c) dan 3(d). Tampak citra hasil enkripsi sudah tidak dapat dikenali lagi karena terlihat seperti citra acak. Dekripsi terhadap *cipher-image* menghasilkan kembali tepat seperti citra 3(a) dan 3(b) semula.



Gambar 3. (a) dan (b) *plain-images*, (c) dan (d) *cipher-images*

Selanjutnya hasil-hasil eksperimen di atas didiskusikan pada bagian di bawah ini, meliputi analisis histogram, analisis hasil iterasi *ACM*, analisis sensitivitas, dan ruang kunci.

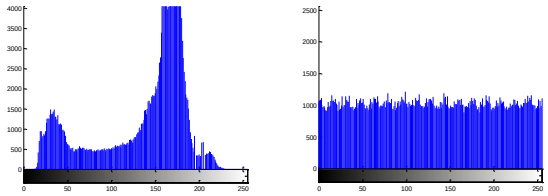
5.1 Analisis Histogram

Histogram memperlihatkan distribusi intensitas *pixel* di dalam citra tersebut. Agar penyerang tidak dapat menggunakan histogram untuk melakukan analisis statistik, maka histogram *plain-image* dan histogram *cipher-image* seharusnya berbeda secara signifikan dan secara statistik tidak memiliki kemiripan. Oleh karena itu, histogram *cipher-image* seharusnya relatif datar (*flat*) atau secara statistik memiliki distribusi (relatif) *uniform* agar menyulitkan penyerang menganalisis frekuensi kemunculan intensitas *pixel* untuk mendeduksi kunci

 ISSN: 1907-5022

atau menemukan *plain-image*. Distribusi yang (relatif) *uniform* pada *cipher-image* adalah sebuah indikasi bahwa algoritma enkripsi citra memiliki tingkat keamanan yang bagus (Jolfaei, 2010).

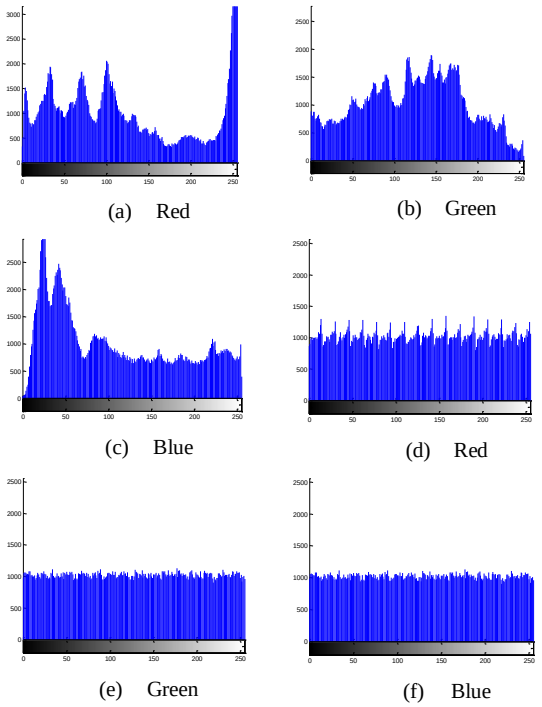
Gambar 4(a) memperlihatkan histogram citra 'kapal' sebelum dienkripsi, dan Gambar 4(b) adalah histogram *cipher-image*-nya. Histogram *cipher-image* terlihat datar dan berbeda secara signifikan dengan histogram *plain-image*.



(a) (b)

Gambar 4. (a) Histogram citra 'kapal' (*plain-image*) dan (b) histogram *cipher-image*.

Gambar 5(a) sampai 5(c) memperlihatkan histogram citra 'bunga' (*plain-image*) untuk setiap kanal warna *RGB* dan Gambar 5(d) sampai 5(f) adalah histogram masing-masang kanal warna pada *cipher-image*. Sama seperti citra 'kapal', histogram *cipher-image* pada setiap kanal *RGB* juga terlihat *flat* atau terdistribusi *uniform*.



(a) Red (b) Green

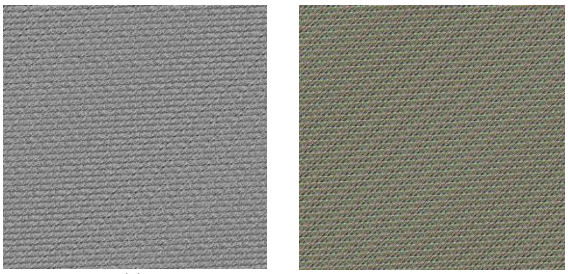
(c) Blue (d) Red

(e) Green (f) Blue

Gambar 5. (a)-(c) Histogram citra 'bunga' (*plain-image*) untuk masing-masing kanal *RGB*; dan (d)-(f) histogram *cipher-image* untuk setiap kanal.

5.2 Analisis Hasil Iterasi *ACM*

Sebelum dienkripsi, citra diacak dengan mengiteraskan *ACM* sebanyak lima kali. Hasil pengacakan *pixel-pixel* citra diperlihatkan pada Gambar 5(a) dan 5(b), masing-masing untuk citra 'kapal' dan citra 'bunga'. Iterasi sebanyak lima kali sudah dapat membuat citra tidak dapat dikenali lagi. Meskipun demikian pengacakan ini tidak mengubah nilai-nilai *pixel* sehingga masih belum aman secara kriptografis. Histogramnya tetap sama seperti pada Gambar 4(a) dan 5(a)-(c).

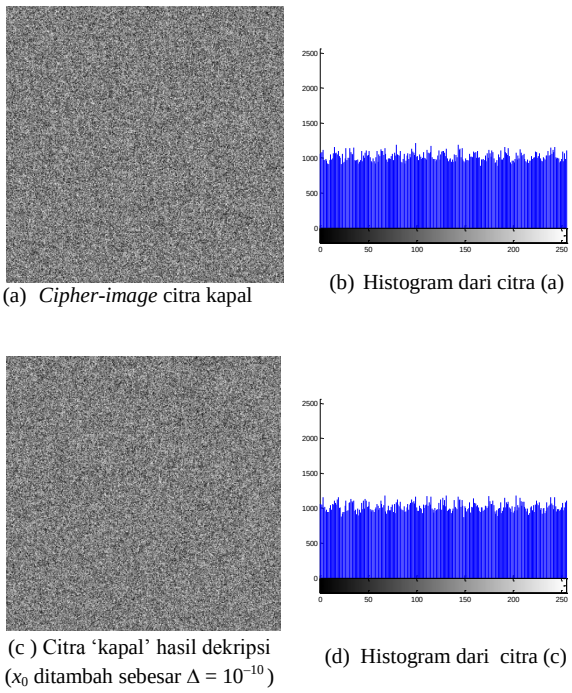


(a) (b)

Gambar 6. (a) dan (b) adalah citra hasil pengacakan dengan iterasi ACM sebanyak m = 5 kali untuk masing-masing citra *plain-images*.

5.3 Analisis Sensitivitas

Parameter nilai awal fungsi *chaos* berperan sebagai (salah satu) kunci rahasia. Sifat *chaos* adalah sensitif terhadap perubahan kecil nilai awal. Sensitif berarti jika nilai kunci diubah sedikit saja maka hasil dekripsi terhadap *cipher-image* menghasilkan *cipher-image* lain yang berbeda.



(a) *Cipher-image* citra kapal (b) Histogram dari citra (a)

(c) Citra 'kapal' hasil dekripsi (x_0 ditambah sebesar $\Delta = 10^{-10}$) (d) Histogram dari citra (c)

Gambar 7. Hasil eksperimen dekripsi dengan pengubahan x_0 sebesar $\Delta = 10^{-10}$.

Pada eksperimen ini nilai awal *logistic map* diubah sebesar Δ sehingga menjadi $x_0 + \Delta$, kemudian citra didekripsi dengan kunci $x_0 + \Delta$. Misalkan $\Delta = 10^{-10}$ sehingga nilai awal *logistic map* menjadi 0.6938000001. Gambar 7 memperlihatkan hasil dekripsi terhadap *cipher-image* dari citra 'kapal'. Hasilnya adalah *cipher–image* lain yang ternyata tetap teracak (tidak kembali menjadi citra semula). Perubahan kecil nilai awal *chaos* membuat nilai acak yang dihasilkan berbeda signifikan setelah fungsi *chaos* diiterasi sejumlah kali. Penyerang yang melakukan *brute force attack* untuk menemukan kunci akan frustasi karena perubahan sangat kecil pada kunci menyebabkan hasil dekripsi tetap salah.

5.4 Ruang Kunci

Ruang kunci menyatakan jumlah total kunci yang berbeda yang dapat digunakan unruk enkripsi/dekripsi (Fu, 2012). Ruang kunci seharusnya cukup besar agar serangan *brute-force attack* menjadi tidak efisien dilakukan. Parameter kunci rahasia yang digunakan di dalam algoritma enkripsi lebih dari satu buah, yaitu p, q, m, x_0, dan r. Tiga parameter pertama, p, q, dan m adalah *integer* positif. Matlab mendukung maksimum *unsigned integer* hingga 32 bit, sehingga nilai pilihan *integer* yang mungkin adalah sekitar $2^{32} = 4.3 \times 10^9$. Untuk nilai awal *Logistic Map* (x_0), presisi komputasi untuk *double-pecision* 64-bit menurut standard *floating-point IEEE* adalah 10^{-15} (Fu, 2012), sehingga jumlah kemungkinan nilai x_0 adalah 10^{15}. Dengan demikian, ruang kunci seluruhnya adalah

$$H(p, q, m, x_0, r) \approx (4.3 \times 10^9) \times (4.3 \times 10^9) \times (10^{15}) \times (10^{15}) \approx 18.49 \times 10^{48}$$

yang cukup besar bertahan terhadap serangan *brute-force attack.*

6. KESIMPULAN

Di dalam makalah telah disajikan sebuah usulan algoritma enkripsi citra digital yang menggabungkan pengunaan dua buah *chaos map* dan teknik enkripsi selektif. *Arnold Cat Map* diiterasikan pada *plain-images* sebanyak *m* kali sebelum dienkripsi secara selektif dengan keystream yang dibangkitkan dengan *Logistic Map*. Dengan teknik enkripsi selektif maka dari setiap *pixel* hanya dienkripsi 4-bit *MSB* saja. Jadi, hanya 50% saja dari keseluruhan citra yang diproses untuk memperoleh citra terenkripsi secara keseluruhan.

Hasil eksperimen memperlihatkan algoritma ini dapat mengenkripsi sembarang citra (baik citra grayscale maupun citra berwarna) dengan baik. *Pixel-pixel* di dalam *cipher-image* mempunyai distribusi relatif *uniform*, hal ini diperlihatkan dengan bentuk histogramnya yang relatif datar, sehingga menyulitkan penyerang melakukan serangan dengan menggunakan analisis statistik.

Eksperimen dengan mengubah sedikit nilai awal *chaos* memperlihatkan bahwa algoritma ini sensitif terhadap perubahan kecil pada kunci sehingga aman dari serangan *exhaustive-key search attack.*

Ruang kunci yang cukup besar membuat algoritma ini tahan terhadap serangan *brute-force attack.*

ACKNOWLEDGMENT

Penelitian yang dipublikasikan di dalam makalah ini sepenuhnya didukung oleh dana **Riset dan Inovasi KK 2012** (Program Riset ITB 2012).

PUSTAKA

Fu, C., Chen, J., Zou, H., Meng, W., Zhan, Y., Yu, Y. (2012), A Chaos-based Digital Image Encryption Scheme with an improved Diffusion Strategy, *Journal Optic Express 2363, Vol. 20. No. 3*.

Jolfaei, A., Mirghadri, A. (2010), An Image Encryption Approach Using Chaos and Stream Cipher, *Journal of Theoretical and Applied Information Technology*.

Lampton, J. (2002), *Chaos Cryptography: Protecting data Using Chaos*, Missisippi School for Mathematics and Science.

Schneier, B. (1996), *Applied Cryptography 2nd Edition*, Wiley & Sons.

Struss, K. (2009), A Chaotic Image Encryption, *Mathematics Senior Seminar*, 4901, University of Minnesota, Morris.

Xiang, T, Wong, K., dan Liao, X. (2007), Selective Image Encryption Using a Spatiotemporal Chaotic System, *Chaos Volume 17*.

Younes, M. A. B, Jantan, A. (2008), Image Encryption Using Block-based Transformation Algorithm, *IAENG International Journal of Computer Science, 35: 1, IJCS_32_1_03*.

Yu, X., Zhang, J., Ren, H., Xu, G., dan Luo, X. (2006), Chaotic Scrambling Algorithm Based on S-DES, *Journal of Physics: Conference Series 48*, 349-353.